An $O(n^2(m + n \log n) \log n)$ Min-Cost Flow Algorithm

Zvi Galil, Eva Tardos

Technical Report
CUCS-302-87

AN $O(n^2(m + n \log n) \log n)$ MIN-COST FLOW ALGORITHM[1]

Zvi Galil[2]

Department of Computer Science

Columbia University

and

Tel-Aviv University

Éva Tardos[3]

Eötvös University, Budapest

and

MSRI, Berkeley

[1] A preliminary version of the paper appeared in Z. Galil and E. Tardos, An $O(n^2(m + n \log n) \log n)$ min-cost flow algorithm. Proc. 27th IEEE Symposium on Foundations of Computer Science, pp. 1-9, October 1986.
[2] Research supported in part by NSF Grants MCS-83-03139 and DCR-85-11713.
[3] Research supported in part by NSF Grant No. 8120790.

Abstract

The minimum-cost flow problem is the following: given a network with n vertices and m edges, find a maximum flow of minimum cost. Many network problems are easily reducible to this problem. A polynomial-time algorithm for the problem has been known for some time, but only recently a strongly polynomial algorithm was discovered.

In this paper we design an $O(n^2(m + n\log n)\log n)$ algorithm. The previous best algorithm had an $O(m^2(m+n\log n)\log n)$ time bound. Thus, we obtain an improvement of two orders of magnitude for dense graphs.

Our algorithm is based on Fujishige's algorithm (which is based on Tardos' algorithm. Fujishige's algorithm consists of up to m iterations, each consisting of $O(m\log n)$ steps. Each step solves a single source shortest path problem with nonnegative edge lengths. We modify this algorithm in order to make an improved analysis possible. The new algorithm may still consist of up to m iterations, and an iteration may still consist of up to $O(m\log n)$ steps, but we can still show that the total number of steps is bounded by $O(n^2\log n)$. The improvement is due to a new technique that relates the time spent to the progress achieved.

1. Introduction

The minimum-cost flow problem is the following: given a network with n vertices and m edges, find a maximum flow of minimum cost. We reformulate the problem, stating it in terms of circulations instead of in terms of flow.

Let R be the set of real numbers. We are given a directed graph $G = (V, E)$, *upper* and *lower capacities* $g \in (R \cup +\infty)^E, f \in (R \cup -\infty)^E$ (satisfying $f \leq g$) and *costs* $d \in R^E$. A vector $x \in R^E$ is called a *circulation* if for every (fixed) node $v \in V$ we have

$$\sum(x(e) : e = uv \in E) - \sum(x(e) : e = vu \in E) = 0.$$

A circulation x is *feasible* if it satisfies the $2|E|$ inequalities $f \leq x \leq g$. We call an inequality with finite $f(e)$ or $g(e)$ a *constraint.* If a constraint is satisfied as an equality, we say that it is *tight* at x. The *cost* of a circulation x is

$$dx = \sum(d(e)x(e) : e \in E).$$

The *minimum-cost circulation problem* is to find a feasible circulation with minimum cost.

We denote an instance of the problem by $P(f,g,d)$. We say that $P(f,g,d)$ is *feasible* if there is a feasible circulation x. We call a feasible circulation *optimal* if it has a minimum cost.

Many network optimization problems are special cases of our problem or can be easily reduced to it (see [L]): (1) The min cost flow problem (essentially equivalent to our problem); (2) the max-flow problem; (3) the shortest path problem; (4) the max (weighted or cardinality) matching in bipartite graphs; and (5) the transportation problem.

An algorithm that solves a problem whose input consists of n real numbers is *strongly polynomial* if (a) it performs only elementary arithmetic operations (additions, subtractions, comparisons, multiplications and divisions); (b) the number of steps is polynomially bounded in n; and (c) when applied to rational data, the size of the numbers (= the number of bits in their representation) that the algorithm generates is polynomially bounded in n and the size of the input numbers. There are several known strongly polynomial algorithms for problems (2), (3) and (4) above. In comparison, an algorithm that solves a problem whose input is a binary string of length L is polynomial (in the usual sense) if its time is bounded by a polynomial in L. The strongly polynomial algorithms mentioned in this paper do not perform multiplications or divisions and consequently they satisfy (c) automatically.

There are polynomial algorithms for the general LP (linear programming) problem ([Kh], [Ka]). Neither algorithm is strongly polynomial. A fundamental open problem is whether the general LP problem can be solved by a strongly polynomial algorithm.

We assume that G has n vertices and m edges. We denote $S(m,n)$, the time needed to solve a single source shortest path problem on G with nonnegative edge lengths. The best algorithm known yields $S(m,n) = O(m + n \log n)$ [FT].

A polynomial-time algorithm for our problem has been known for some time [EK]. The algorithm of Edmonds and Karp, the Out of Kilter algorithm with scaling (what we will call the EK-algorithm), takes time $O(mS(m,n)\log M)$ where M is the maximum among the finite values of $|f(e)|$ and $|g(e)|$ for $e \in E$. The first strongly polynomial algorithm was found only recently [Ts].

The strongly polynomial algorithm was based on the following observation: Let P be our problem and assume it is feasible (since it is easy to test feasibility). Assume we scale the capacities to a small interval ("small" means polynomial in m and n) and then round each capacity to a nearby integer. The rounding is done in such a way that the new problem, P^*, is still feasible. (Namely, upper capacities are rounded up and lower capacities down.) Using the EK-algorithm, we solve P^* in time $O(mS(m,n)\log n)$. Now we scale the solution back and assume we obtain x^*. Let x be an optimal solution of P closest to x^*. One then shows that for any constraint in P^* that is "far" from being tight at x^*, the corresponding constraint in P cannot be tight at x. Thus, by deleting this constraint (by setting $f(e)$ to $-\infty$ or $g(e)$ to $+\infty$), one obtains a problem P' for which x is still optimal. If each iteration deletes at least one constraint, then after at most $2m$ such iterations we derive a problem $\bar{P}$ such that every feasible solution of $\bar{P}$ is optimal.

It remains to guarantee that indeed we can relax at least one constraint; i.e. that at least one constraint is far enough. Here, one observes that the problem is not changed if we change the origin. So we can subtract from f and g a circulation x_0.

During each iteration, Tardos used projection (or solving a system of linear equations) to find an origin which guarantees the deletion of at least one constraint. As a result, the task of finding the origin became more expensive than the execution of the EK-algorithm in each iteration.

Fujishige chooses a different origin. He constructs a spanning tree T that contains a maximum number of edges with both upper and lower capacities infinite. He chooses a circulation x_0 so that many of the finite capacities become zero: for $e \notin T$ $x_0(e)$ is chosen to be zero if both capacities are infinite, and $x_0(e)$ is chosen to be either $f(e)$ or $g(e)$ otherwise; then x_0 is completed in T to make it a circulation. One can easily show that this x_0 is good enough for our purpose of deleting at least one constraint. Since x_0 is found in time $O(m)$, the dominating part of an iteration is now the execution of the EK-algorithm (on P^*).

One way of improving the time bound of an algorithm is to prove that it makes more progress in every iteration. Sometimes one needs to modify the algorithm before such a proof is possible. Dinic's network flow algorithm reduced the number of iterations from m to n by finding many augmenting paths simultaneously [D]. Hopcroft and Karp's bipartite matching algorithm improved the number of iterations from n to $O(\sqrt{n})$ in a similar way [HK]. Karmarkar's algorithm for LP [Ka] seems to perform in practice many fewer iterations than can be proved. It is a challenge to prove a better bound, possibly by modifying the algorithm.

In our case, we could not prove that the algorithm performs fewer iterations, because we are not able to guarantee that more than one constraint will be deleted per iteration. Instead, we modified the algorithm in several places so that we could prove that the total cost of all the iterations is decreased, by relating the progress achieved to the time spent.

The EK-algorithm consists of steps. Each step solves a single source shortest path problem with nonnegative edge lengths. (See the Appendix for the EK-algorithm.) The number of steps in the general EK-algorithm is $O(m \log M)$ ($= O(m \log n)$ in our case). Thus, the number of steps in Fujishige's algorithm is $O(m^2 \log n)$. We define a function $F(f, g, d)$ with values bounded by $O(n^2)$. We redesign the algorithm in such a way that if an iteration performs $N \log n$ steps, then the value of F decreases by N. Consequently, the total number of steps is at most $O(n^2 \log n)$.

In Section 2 we sketch some facts needed later, in Section 3 we present the algorithm, in Section 4 we prove its validity, and in Section 5 we prove its run time.

2. Preliminaries

In this section we consider $P(f, g, d)$. For a positive integer k let $E_k(f, g)$ be the set of edges e such that at least one of its capacities is finite and its absolute value is at least M/k, where M is as defined above. Note that $E_1 \subseteq E_2 \subseteq \cdots$ We define $E_\infty(f, g) = \{e \in E \text{ with } g(e) = -f(e) = \infty\}$.

For a function h let $E(h) = \{e \in E \text{ with finite } h(e)\}$. We call any function $p : V \to R$ a potential and denote by $d_p : E \to R$ the function defined by $d_p(uv) = d(uv) + p(u) - p(v)$. The following lemma states the complementary slackness principle of LP for our problem [FF].

Lemma 2.1. There exists a potential p such that a feasible circulation x is minimum cost if and only if x satisfies

$$(*)\ d_p(e) > 0 \Rightarrow x(e) = f(e) \text{ and } d_p(e) < 0 \Rightarrow x(e) = g(e).$$

We call a potential p satisfying $(*)$ *optimal.*

Lemma 2.2. For any potential p, a feasible circulation is minimum cost subject to the cost function d if and only if it is minimum cost subject to d_p.

The basic ingredient of our algorithm is the following slight generalization of a lemma by Fujishige. It states when a constraint can be relaxed.

Lemma 2.3. Let $\epsilon > 0$ and let f^* and g^* be new lower and upper capacities. Suppose that $E(f^*) = E(f)$ and $E(g^*) = E(g)$ and $|f(e) - f^*(e)| < \epsilon$ for $e \in E(f)$ and $|g(e) - g^*(e)| < \epsilon$ for $e \in E(g)$. Further suppose that $P(f, g, d)$ is feasible and that x^* is an optimal solution of $P(f^*, g^*, d)$. Then there is an optimal solution x of $P(f, g, d)$, such that for every $e \in E$

$$|x^*(e) - f^*(e)| \geq \epsilon|E| \Rightarrow x(e) > f(e) \text{ and } |x^*(e) - g^*(e)| \geq \epsilon|E| \Rightarrow x(e) < g(e).$$

Proof: The lemma is a special case of Theorem 5 in [CGST]. ▮

Relaxing constraints that are not tight at an optimal solution does not change the set of optimal potentials:

Lemma 2.4. Let x be a minimum-cost circulation in $P(f,g,d)$. Let E_1 and E_2 be subsets of E such that $x(e) > f(e)$ for $e \in E_1$ and $x(e) < g(e)$ for $e \in E_2$. Define $f^*(e) = -\infty$ for $e \in E_1$ and $f(e)$ otherwise, and $g^*(e) = +\infty$ for $e \in E_2$ and $g(e)$ otherwise. Then the problems $P(f,g,d)$ and $P(f^*,g^*,d)$ have the same sets of optimal potentials.

Proof: Let p be a potential and x' be a feasible circulation for $P(f,g,d)$. If p and x' satisfy condition $(*)$ of Lemma 2.1 then, by complementary slackness, x' is a minimum-cost circulation and p is an optimal potential.

Now let p be an optimal potential for $P(f,g,d)$. By definition p and the minimum-cost circulation x satisfy $(*)$. Furthermore, by the conditions of the lemma, p and x also satisfy $(*)$ with f replaced by f^* and g replaced by g^*. Thus x is a minimum-cost circulation and p an optimal potential for $P(f^*,g^*,d)$.

To see the reverse inclusion let p be an optimal potential for $P(f^*,g^*,d)$. We just showed that x is a minimum-cost circulation for $P(f^*,g^*,d)$, and so p and x have to satisfy $(*)$ with f^* and g^*. Then p and x satisfy $(*)$ also with f and g , and so p is an optimal potential for $P(f,g,d)$. ▮

Our algorithm will round to integers differently than was done in previous algorithms: positive numbers will be rounded down and negative numbers will be rounded up. Such a rounding guarantees that capacities with small absolute values will be rounded to zero and not to 1 or -1, which will have an important effect on the time analysis. To maintain feasibility, we solve the new problem on a new graph, $G' = (V', E')$ that is an extension of $G : V' = V \cup \{s\}, E' = E \cup \{sv, vs \text{ for } v \in V\}$ and the new edges will have zero lower capacities, infinite upper capacities, and effectively infinite costs.

3. The algorithm

Step 0 (initialization):

—Check whether there exists a feasible flow. (This can be done by solving a max-flow problem [FF].) If not STOP.

—Put $f' = f$, $g' = g$. Find a potential p with

$$d_p(e) \le 0 \text{ for } e \in E \text{ with } f'(e) = -\infty \text{ and } d_p(e) \ge 0 \text{ for } e \in E \text{ with } g'(e) = +\infty.$$

(See Lemma 4.1.) If there is no such p, STOP. (The minimum cost is $-\infty$.)

Step 1: (Steps 1-3 constitute an iteration.)

—Define the following weights on the edges

$$w(e) = \begin{cases} g'(e) - f'(e), & \text{if both } f'(e) \text{ and } g'(e) \text{ are finite and} \\ 0 & \text{otherwise.} \end{cases}$$

—Find a maximum weight spanning tree T of G among trees with maximum number of edges in $E_\infty(f', g')$. (This task is easily reduced to that of finding a maximum spanning tree without the restriction.)

— Find a circulation x' (not necessarily feasible) such that $x'(e) = 0$ for all $e \in E_\infty(f', g')\backslash T$ and $x'(e) = f'(e)$ or $g'(e)$ (one of the two which is finite) for all $e \in E\backslash(T \cup E_\infty(f', g'))$. (See Lemma 4.3)

—Put $f''(e) = f'(e) - x'(e)$ and $g''(e) = g'(e) - x'(e)$ for all $e \in E$.

— Put $M = \max(0, \max|g''(e)| \text{ for } e \in E(g'')), \max(|f''(e)| \text{ for } e \in E(f'')))$.

— If $M = 0$ go to Step 4, otherwise proceed to Step 2.

Step 2 (defining the rounded problem):

— Let r be the smallest integer power of 2 greater or equal $2(m + 2n)^2$ and let k be the smallest integer power of r such that $|E_{rk}(f'', g'')| \le 2|E_k(f'', g'')|$. (See Lemma 4.2) (The choice of k determines the unit $(= M/(rk))$ in the rounded problem. It is chosen so that in the rounded problem the number of edges with at least one nonzero finite capacity is $O(|E_k(f'', g'')|)$.)

— For all $e \in E$ round $f''(e)rk/M$ and $g''(e)rk/M$ to the nearest integer below or above, rounding positive numbers up and negative numbers down. Let $\bar{f}(e)$ and $\bar{g}(e)$ be the resulting integers respectively.

— Extend $\bar{f}$ and $\bar{g}$ to G' by $\bar{f}(sv) = \bar{f}(vs) = 0$ and $\bar{g}(sv) = \bar{g}(vs) = +\infty$. Further, put $\bar{d}(e) = d_p(e)$ for $e \in E$ and $\bar{d}(e) = \sum(|d_p(e)| : e \in E)$ for e in $E' \setminus E$. (Note that $P(\bar{f}, \bar{g}, \bar{d})$ is feasible, since any vector $x \in R^E$ such that $\bar{f} \le x \le \bar{g}$ can be extended to $x' \in R^{E'}$ which is a feasible circulation of $P(\bar{f}, \bar{g}, \bar{d})$.)

Step 3 (solving the rounded problem and relaxing constraints):

— Using the EK-algorithm, find a minimum-cost circulation $\bar{x}$ and an optimal potential $\bar{p}$ for $P(\bar{f}, \bar{g}, \bar{d})$. Use the modified rounding, that is, rounding positive numbers down and negative numbers up, inside the EK-algorithm too. See the Appendix for this version of the EK-algorithm.

— For all $e \in E$ if $\bar{g}(e) - \bar{x}(e) \ge m + 2n$ put $g'(e) = +\infty$, and if $\bar{x}(e) - \bar{f}(e) \ge m + 2n$ put $f'(e) = -\infty$.

— Put $p = \bar{p}$ and go to Step 1.

Step 4 (finding the optimal circulation, the potential p is optimal already):

— Set

$$f'(e) = \begin{cases} f(e) & \text{if } d_p(e) \ge 0; \text{ and} \\ g(e) & \text{otherwise} \end{cases}$$

and

$$g'(e) = \begin{cases} g(e) & \text{if } d_p(e) \le 0; \text{ and} \\ f(e) & \text{otherwise} \end{cases}$$

—Find a feasible circulation x in $P(f', g', d)$, and output x as a minimum-cost circulation.

4. Validity

Lemma 4.1. The potential required in Step 0 can be found via a shortest path computation.

Proof: Define the following length function:

$$\ell(uv) = \begin{cases} -d(e) & \text{if } e = uv \text{ and } f'(e) = -\infty \\ d(e) & \text{if } e = vu \text{ and } g'(e) = +\infty. \end{cases}$$

Now consider the graph (V, E_1), where $E_1 = \{uv$ such that either $uv \in E$ and $f'(uv) = -\infty$ or $vu \in E$ and $g'(vu) = +\infty\}$. If (V, E_1) contains a negative length cycle, then there is no minimum cost circulation (since any feasible circulation can be improved along the cycle). Otherwise, the distances from a fixed vertex s give the required potential. ▮

Lemma 4.2. The value k chosen in Step 2 satisfies $\log k = O(\log^2 n)$.

Proof: Let $r = 2(m+2n)^2$ and $k = r^l$ as in the algorithm. By definition $|E_1(f'', g'')| \geq 1$. By the minimal choice of k if $l' < l$ then

$$|E_{r^{l'+1}}(f'', g'')| > 2|E_{r^{l'}}(f'', g'')|.$$

Thus $|E_k(f'', g'')| > 2^l$, and so $l \leq \log m$ and $\log k \leq O(\log m \ \log r)$. ▮

Lemma 4.3. The circulation x' specified in Step 1 of the algorithm can be found in $O(m)$ time.

Proof: Set x' to be equal to the specified value on the edges outside the chosen tree T. We can extend x' to the edges of the tree by iteratively balancing the flow at a leaf of the tree and deleting that node from the tree. ▮

Lemma 4.4. Setting the capacities to infinity in Step 3 does not change the set of optimal potentials.

Proof: From Step 0 we may assume that $P(f, g, d)$ is feasible. Extend f and g to G' by $f(sv) = f(vs) = 0$ and $g(vs) = g(sv) = +\infty$. Let $\bar{d}$ be the cost function defined in Step

2 in the first iteration. Due to the large cost of the edges in $E'\backslash E$, any minimum-cost circulation x for $P(f,g,\bar{d})$ on G' will have zero values on all edges in $E'\backslash E$. Further, by Lemma 2.2, x restricted to E is a minimum-cost circulation for $P(f,g,d)$ on G.

Now consider the first iteration. We have $f' = f$ and $g' = g$. The polyhedron $P(f'',g'',\bar{d})$ is a translation of $P(f,g,\bar{d})$, and thus they have the same set of optimal potentials. Now apply Lemma 2.3 to the problems $P(f'',g'',\bar{d})$ and $P(f^*,g^*,\bar{d})$ where $f^* = M(kr)^{-1}\bar{f}$ and $g^* = M(kr)^{-1}\bar{g}$ (where $\bar{f}$ and $\bar{g}$ are defined in Step 2 in the first iteration). Let $\epsilon = M(kr)^{-1}$. The circulation $x^* = M(kr)^{-1}\bar{x}$ is a minimum-cost circulation for $P(f^*,g^*,\bar{d})$ where $\bar{x}$ is the circulation found in Step 3. Let x be the minimum-cost circulation given by the lemma. As discussed above, $x + x'$ (where x' is the vector defined in Step 1) restricted to E is a minimum-cost circulation for $P(f,g,d)$ on G. So Lemma 2.4 proves that the set of optimal potentials is the same after the first iteration.

Now the conclusion of the lemma follows by induction on the number of iterations.

∎

As a corollary of the above lemma we get

Theorem 4.5. *When the algorithm performs Step 4, the potential p is optimal, and so the circulation x is a minimum-cost solution to $P(f,g,d)$.*

Proof: We first prove that just before the algorithm performs Step 4, the current potential p is optimal for the current $P(f',g',d)$. Since $M = 0$, all finite constraints of $P(f',g',d)$ are tight for the circulation $x = 0$, so it suffices to show that $d_p(e) \geq 0$ for all edges e with $g'(e) = +\infty$ and $d_p(e) \leq 0$ for all edges e with $f'(e) = -\infty$.

Let us prove the latter statement by induction on the number of iterations performed. Initialization insures that the statement is true at the beginning of the first iteration. If $f'(e) = -\infty$ after any iteration, then in this iteration $\bar{x}(e) > \bar{f}(e)$, and so, by complementary slackness (condition $(*)$ of Lemma 2.1), $d_{\bar{p}}(e) \leq 0$. Similarly if $g'(e) =$

$+\infty$ after an iteration then $d_{\bar{p}}(e) \geq 0$ in the previous execution of Step 3.

Now, by Lemma 4.4 p is optimal for $P(f, g, d)$ (the original problem). By Lemma 2.1 a feasible solution of $P(f, g, d)$ is optimal if and only if it satisfies $(*)$, that is if and only if it is a feasible solution of $P(f', g', d)$ with f' and g' defined in Step 4. Thus $P(f', g', d)$ is feasible and the circulation found in Step 4 is an optimal solution for $P(f, g, d)$. ∎

5. Running time

Given $P(f,g,d)$, let

$$F(f,g,d) = |E(f)| + |E(g)| + [\text{comp}(E_\infty(f,g))]^2$$

where $\text{comp}(E_1)$, for a subset of edges E_1, is the number of connected components in the underlying undirected graph (V, E_1). The following lemma helps relating the time spent on an iteration to the progress achieved. Note that $F(f,g,d) \leq 2m + n^2 = O(n^2)$.

Lemma 5.1. The value of $F(f',g',d)$ decreases by at least $|E_k(f'',g'')|$ at each execution of Step 3, where k is the value chosen in Step 2.

Proof: Without loss of generality we may assume that $E(g') \subseteq E(f')$, $x'(e) = f'(e)$ for $e \in E(g')\backslash T$, and $|g'(e) - x'(e)| \leq |f'(e) - x'(e)|$ for $e \in E(g') \cap T$. (We can reverse all edges not satisfying the above assumptions.) The following properties, that are easy consequences of the choice of T and the above assumptions, are repeatedly used below.

—If an edge e is in $T\backslash E_\infty(f',g')$ then all other edges in the cut defined by T and e also have at least one finite capacity.

— If $e \in E_l(f'',g'')$, $e \notin T$ then $|g''(e)| \geq M/l$.

— If $e \in T \cap E_l(f'',g'')$ then $|f''(e)| \geq M/l$.

— If an edge e is neither in T nor in $E_\infty(f',g')$ then $f''(e) = \bar{f}(e) = 0$.

Consider an edge e_0 in $E_k(f'',g'')$ (where k is the value chosen in Step 2). We would like to conclude that at least one of $f'(e_0)$ and $g'(e_0)$ will be replaced by infinity.

Case 1: $e_0 \notin T$

We have $f''(e_0) = 0$, and so $g''(e_0) \geq M/k$. Thus $|\bar{g}(e_0) - \bar{x}(e_0)| + |\bar{f}(e_0) - \bar{x}(e_0)| \geq r \geq 2(m+2n)^2$ and therefore at least one of $f'(e_0)$ or $g'(e_0)$ will be replaced by infinity in Step 3.

Case 2: $e_0 \in T$

Now $|f''(e_0)| \geq M/k$. The tree T and e_0 define a cut. The edges $e \neq e_0$ crossing the cut must satisfy $\bar{f}(e) = 0$. Thus the sum of the capacities $\bar{f}(e)$ over the cut has absolute value at least $r \geq 2(m+2n)^2$. The sum of the circulation over any cut is zero. The cut contains at most $m + 2n$ edges (of G'). Thus there exist an edge e_1 in the cut with $\bar{x}(e_1) - \bar{f}(e_1) \geq 2(m+2n)$. Now there are three cases

Case 2a: $e_1 = e_0$

The value $f'(e_0)$ is replaced by $-\infty$.

Case 2b: $e_1 \neq e_0$ and $e_1 \in E(g')$.

Let us estimate $w(e_0)$. First, because of the choice of the tree $w(e_0) \geq w(e_1)$. Furthermore

$$\begin{aligned} w(e_1) = g'(e_1) - f'(e_1) = g''(e_1) > M(rk)^{-1}(\bar{g}(e_1) - 1) \geq \\ \geq M(rk)^{-1}(\bar{x}(e_1) - 1) = M(rk)^{-1}(\bar{x}(e_1) - \bar{f}(e_1) - 1) \geq \\ \geq M(rk)^{-1}(2(m+2n) - 1). \end{aligned}$$

Thus we have

$$\begin{aligned} (\bar{g}(e_0) - \bar{x}(e_0)) + (\bar{x}(e_0) - \bar{f}(e_0)) = \bar{g}(e_0) - \bar{f}(e_0) > \\ rkM^{-1}(g''(e_0) - f''(e_0)) - 1 = rkM^{-1}w(e_0) - 1 \geq \\ 2(m+2n) - 2. \end{aligned}$$

Both $\bar{g}(e_0) - \bar{x}(e_0)$ and $\bar{x}(e_0) - \bar{f}(e_0)$ are integers, so at least one of $f'(e_0)$ and $g'(e_0)$ will be replaced by infinity.

Case 2c: $e_1 \neq e_0$ and $g'(e_1) = \infty$.

By the observations above, e_1 connects two distinct connected components of $E_\infty(f', g')$. In this iteration we set $f'(e_1)$ to $-\infty$ and so e_1 is added to $E_\infty(f', g')$. As a result comp$(E_\infty(f', g'))$ will decrease by at least one.

Now we are ready to prove the lemma. If Case 2c does not occur during an iteration, $F(f', g', d)$ clearly decreases by at least $|E_k(f'', g'')|$. Suppose Case 2c does occur. Let $c=$ comp$(E_\infty(f', g'))$. Now $F(f', g', d)$ decreases by at least

$$|E_k(f'', g'')\backslash T| + c^2 - (c-1)^2 = |E_k(f'', g'')| - |E_k(f'', g'') \cap T| + 2c - 1 \geq$$

$$|E_k(f'', g'')| - |E(f'') \cap T| + 2c - 1 = |E_k(f'', g'')| - (c-1) + 2c - 1 \geq |E_k(f'', g'')|. \quad ∎$$

The following corollary will be useful in bounding the number of arithmetic operations performed in Steps 1 and 2.

Corollary 5.2. The number of iterations is at most $2m$.

Proof: At least one capacity is replaced by infinity in each iteration. ∎

Consider an execution of the EK-algorithm during our algorithm. Because of the choice of k and the different rounding we have

Lemma 5.3. The EK-algorithm in Step 3 consists of $O(|E_k(f'', g'')| \log n)$ executions of a single source shortest path subroutine.

Proof: The number of executions of the single source shortest path subroutine during the EK-algorithm is the the number of ones in the binary representation of the finite capacities. (See the Appendix for an analysis of the EK-algorithm.) In our case the number of edges with at least i bits in the present iteration is at most $|E_{2^i}(f'', g'')|$. Recall that $k = r^l$. Now the sum of the kilter numbers can be estimated as

$$\sum_{j=0}^{\log rk} |E_{2^j}(f'', g'')| \leq \log r \sum_{j=0}^{l+1} |E_{r^j}(f'', g'')| \leq \log r(2|E_k(f'', g'')| + |E_{kr}(f'', g'')|)$$

$$\leq 4|E_k(f'', g'')| \log r = O(|E_k(f'', g'')| \log n).$$

The first inequality is due to the fact that $|E_i(f'', g'')| \subseteq |E_{i+1}(f'', g'')|$, $i = 0, 1, \ldots$ The last two inequalities are due to the choice of k. ∎

Theorem 5.4. *The algorithm uses $O(n^2(m + n\log n)\log n)$ arithmetic operations.*

Proof: Step 0 and Step 4 consist of solving two maximum flow and one shortest path problem in time $O(mn\log n)$ [Tn]. By Corollary 5.2 we know that Step 1 is executed at most $2m$ times, each time computing maximum weight spanning tree in time $O(m\log n)$ [Tn] for a total of $O(m^2\log n)$. The only non-trivial part in Step 2 is finding $\bar{f}$ and $\bar{g}$ since rounding is not on our list of elementary arithmetic operations. We first check for each edge e if either $\bar{f}(e)$ or $\bar{g}(e)$ is finite and nonzero (if $e \in E_{rk}(f'', g'')$) and if it is we compute $\bar{f}(e)$ and $\bar{g}(e)$ by using binary search. So Step 2 takes $O(m + |E_k(f'', g'')|\log rk)$ by the choice of k and since the finite absolute values of $\bar{f}$ and $\bar{g}$ are bounded by rk. By Corollary 5.2 the first term sums up to $O(m^2)$ and by Lemma 5.1 and Lemma 4.2 the second term sums up to $O(n^2\log^2 n)$.

By Lemma 5.1, the total number of executions of the single source shortest path subroutine is at most $O(n^2\log n)$. Therefore, the total cost of Step 3 is $O(n^2(m + n\log n)\log n)$. This is the cost of the algorithm since the other parts of the algorithm contribute less to the total time. ∎

6. Appendix

We summarize a version of the EK-algorithm. We present an analysis slightly more refined than that of Edmonds and Karp [EK]. The algorithm is given in a form similar to that found in Lawler's book [L]. The algorithm given in Lawler's book is incomplete and seems to implicitly assume a linear time single-source shortest path subroutine.

Our version of the EK-algorithm is based on the following version of the Out-of-Kilter method [FF]. Let $P_G(f, g, d)$ denote the minimum cost circulation problem on a graph G with integral (or infinite) capacities f and g. Let x be an integral circulation on G (not necessarily feasible), and p an arbitrary potential. We define the *kilter number* of an edge e subject to x and p as

$$k(e) = \begin{cases} |f(e) - x(e)| & \text{if } d_p(e) > 0 \text{ or } d_p(e) = 0 \text{ and } x(e) < f(e) \\ |g(e) - x(e)| & \text{if } d_p(e) < 0 \text{ or } d_p(e) = 0 \text{ and } x(e) > g(e) \\ 0 & \text{if } d_p(e) = 0 \text{ and } f(e) \le x(e) \le g(e). \end{cases}$$

The kilter number of x and p is $k(x, p) = \sum_{e \in E} k(e)$. Observe that x and p are optimal if and only if $k(x, p) = 0$.

Define the *residual graph* G_x on the vertex set of G with edges

$$E_x = \{uv : \text{either } e = uv \in E \text{ and } x(e) < g(e) \text{ or } e = vu \in E \text{ and } x(e) > f(e)\}.$$

Furthermore, define a cost function $\tilde{d}$ on E_x as

$$\tilde{d}(uv) = \begin{cases} d_p(e) & \text{if } e = uv \in E \\ -d_p(e) & \text{if } e = vu \in E. \end{cases}$$

The following procedure due to Edmonds and Karp [EK] reduces the kilter number, and can be implemented using one call to a single-source shortest path subroutine (with non-negative edge lengths).

Kilter Number Reduction

1. Define the length function $\ell(uv) = \max(0, \tilde{d}(uv))$ for edges $e \in E_x$.

2. Choose u^*v^* such that

(i) $d_p(u^*v^*) < 0$ and $x(u^*v^*) < g(u^*v^*)$ or

(ii) $d_p(v^*u^*) > 0$ and $x(v^*u^*) > f(v^*u^*)$ or

(iii) $x(u^*v^*) < f(u^*v^*)$ or

(iv) $x(v^*u^*) > g(v^*u^*)$,

that is, the corresponding edge has a positive kilter number.

3. Let $N^* = \{v | v = v^*$ or G_x has a directed path from v^* to $v\}$. For $v \in N^*$ let $\delta(v)$ equal the minimum length (ℓ) of a directed path from v^* to v. For $v \notin N^*$ let $\delta(v) = \max\{\delta(u) - \tilde{d}(wu) : \; wu \in E_x, \; u \in N^*, w \notin N^*\}$.

4. Let $p' = p + \delta$.

— If there is an edge $uw \in E$ such that either $u \in N^*$ and $w \notin N^*$ and $x(uw) > g(uw)$ or $w \in N^*$ and $u \notin N^*$ and $x(uw) < f(uw)$, then STOP, there is no feasible circulation.

— If $u^* \in N^*$, choose a minimum length path P from v^* to u^* in G_x. If $P = v^*u^*$ let $x' = x$. Otherwise, augment the circulation by one along the cycle consisting of P and u^*v^* and let x' be the resulting circulation.

— Otherwise ($u^* \notin N^*$ and u^*v^* was chosen according to options (i) or (ii) in Step 2) $x' = x$.

Lemma 6.1. The procedure Kilter Number Reduction either stops announcing correctly that there is no feasible circulartion or reduces the kilter number $k(x,p)$.

Proof Sketch: The correctness of announcing infeasibility is left to the reader. We assume that the procedure does not stop and show that the kilter number of an edge never increases and the kilter number of the edge corresponding to the vertex pair u^*v^* decreases. First consider the cut defined by the vertex set N^*. Due to the definition of the residual graph G_x and not stopping in the first test of Step 4, we have that for

$uw \in E$, $u \in N^*, w \notin N^*$, $x(uw) = g(uw)$ and for $uw \in E$, $w \in N^*, u \notin N^*$, $x(uw) = f(uw)$. By the choice of and p' (i.e., by the large constant added to the potential of all $v \notin N^*$), the kilter number of all the edges with one endpoint in N^* is 0. (This constant guarantees that d_p of such an edge is of the right sign.) Furthermore, observe that the kilter number of the edges with both vertices outside N^* is unchanged. Finally, consider edges with both vertices in N^*. If the circulation is unchanged on the edge, then the kilter number cannot increase, due to the definition of p'. Now suppose that the circulation value is changed on the edge. In this case the edge is on some shortest path from v^*. Consequently, the new changed cost $d_{p'}(e)$ will be zero if the edge is used in a direction in which $\tilde{d} = \ell$. Otherwise, the difference between the circulation value and the appropriate capacity decreases. In other words, the kilter number cannot increase. By considering the same cases more carefully, we see that the kilter number of the edge corresponding to the vertex pair u^*v^* decreases. ∎

Next we describe the EK-algorithm. It solves a series of approximated problems. We use the modified rounding which does not maintain feasibility. To preserve feasibility, we use the extended graph G' as we did in Step 2 of the main algorithm: $G' = (V', E')$, where $V' = V \cup \{s\}$, $E' = E \cup \{vs, sv \text{ for } v \in V\}$. We extend the capacities and the costs to E' by defining, for each $v \in V$, $f(sv) = f(vs) = 0$, $g(sv) = g(vs) = +\infty$, $d(sv) = d(vs) = \sum(|d(e)| \text{ for } e \in E)$. Observe that this extended problem is feasible and it has a feasible dual solution if and only if the original one has. In fact, when using the EK-algorithm in Step 3 of the main algorithm the extension of the graph is not necessary, since we are already working on an extended graph. Now we are ready to describe the algorithm.

First check whether or not the original problem is feasible (using a maximum flow computation). Let μ denote the maximum binary length of the capacities. We define the approximated problems on the extended graph G' by $P_{G'}(f_i, g_i, d)$ with capacities f_i and g_i obtained from $f/2^{\mu+1-i}$ and $g/2^{\mu+1-i}$ by rounding positive numbers down and negative numbers up. Observe that, due to the definition of G', each of the approximated problems is feasible, and an optimal solution to $P_{G'}(f_{\mu+1}, g_{\mu+1}, d)$ restricted to the edges in E gives an optimal solution to the original problem. We solve $P_{G'}(f_i, g_i, d)$ in order $i = 0, 1, \ldots, \mu + 1$.

First consider $i = 0$. Each capacity is either zero or infinity. Let p_0 denote a feasible dual solution to $P_{G'}(f_{\mu+1}, g_{\mu+1}, d)$. Clearly $x_0 = 0$ and p_0 are optimal primal and dual solutions of $P_{G'}(f_0, g_0, d)$.

Now suppose x_i and p_i are optimal primal and dual solutions for $P_{G'}(f_i, g_i, d)$, $i \geq 0$. Consider the kilter numbers of the pair p_i and $2x_i$ for the problem $P_{G'}(f_{i+1}, g_{i+1}, d)$. The kilter number of each edge is at most one. Apply the above Kilter Number Reduction procedure to get an optimal solution to the $(i+1)$-th problem.

Theorem 6.2[EK]: The minimum cost circulation problem can be solved using $O(|E|\mu)$ calls to a subroutine for finding single source shortest path with non-negative edge lengths.

For the time analysis we need a slightly stronger version of Theorem 6.2. The edge e can have a non-zero kilter number subject to $2x_{i-1}$ and p_{i-1} in the i-th approximated problem only if $f_i(e) \neq 2f_{i-1}(e)$ or $g_i(e) \neq 2g_{i-1}(e)$. That is, the i-th bit of either $f(e)$ or $g(e)$ is one. This yields the following strengthening of Theorem 6.2.

Theorem 6.3: The minimum cost circulation problem can be solved using one shortest path computation for a graph without negative cycles, and as many computations of single source shortest path with non-negative edge lengths as the number of ones in the binary representations of the capacities.

Conclusion

In this paper we designed an $O(n^2(m + n\log n)\log n)$ algorithm for the min-cost flow problem. The algorithm is a modification of Fujishige's algorithm. The modifications made it possible to relate the progress achieved to the time spent. The bound improves the previous best bound whenever $m = \omega(n)$. The improvement is the largest (by a factor of $O(n^2)$) for dense graphs ($O(n^4 \log n)$ compared to $O(n^6 \log n)$).

Acknowledgements

We would like to thank Dorit Hochbaum and David Shmoys for careful reading of the manuscript and for their valuable help in improving the presentation. We would also like to thank Stuart Haber for his helpful comments.

References

[CGST] W. Cook, A. M. H. Gerards, A. Schrijver and E. Tardos, Sensitivity theorems in integer linear programming. *Math. Programming 34* (1986), 251–264.

[D] E. A. Dinic, Algorithm for solution of a problem of maximal flow in a network with power estimation. *Soviet Math. Dokl. 11* (1970), 1277–1280.

[EK] J. Edmonds and M. R. Karp, Theoretical improvements in the algorithmic efficiency for network flow problems, *J. ACM 19* (1972), 248–264.

[F] S. Fujishige, An $O(m^3 \log n)$ capacity-rounding algorithm for the minimum-cost circulation problem: a dual framework of the Tardos algorithm, *Mathematical Programming 35* (1986), 298–308.

[FF] L. R. Ford and D. R. Fulkerson, *Flows in Networks.* Princeton University Press, Princeton, N.J., 1962.

[FT] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses. *Proc. 25-th Annual IEEE Symp. on Foundations of Computer Science,* 1984, 338–346.

[HK] J. E. Hopcroft and R. M. Karp, $N^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. on Comput. 2* (1973), 225–231.

[Ka] N. Karmarkar, A new polynomial-time algorithm for linear programming. *Combinatorica 4* (1984), 373–395.

[Kh] L. G. Khachiyan, A polynomial algorithm in linear programming. *Soviet Math. Dokl. 20* (1979), 191–194.

[L] E. L. Lawler, *Combinatorial Optimization, Networks and Matroids.* Holt, Rinehart and Winston, N. Y., 1976.

[O] J. B. Orlin, Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem. Working paper No. 1615-84, A. P. Sloan School of Management, MIT, December 1984.

[Ts] E. Tardos, A strongly polynomial minimum cost circulation algorithm. *Combinatorica 5* No. 3 (1985), 247–255.

[Tr] R. E. Tarjan, *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, Penn. , 1983.